

**Mobiliya Fleet Management**

**User Guide**

|  |  |
| --- | --- |
| **Revision History** | |
| Version No. | 1.0 |
| Authorized By | Sagar Shah |

© 2018 Mobiliya Technologies

Strictly Private and Confidential. No part of this document should be reproduced or distributed without the prior permission of Mobiliya Technologies.

Table of Contents

[1. Introduction 4](#_Toc9743)

[1.1. About this Guide 4](#_Toc19016)

[1.2. About Mobiliya Fleet Management Solution 4](#_Toc18482)

[2. System Architecture 5](#_Toc866)

[2.1. Mobile App Components 5](#_Toc5364)

[2.2. Cloud Application Components 7](#_Toc12557)

[3. Intended Audience 8](#_Toc23495)

[4. Setup 9](#_Toc18997)

[5. User Instructions 18](#_Toc12300)

[1. Web Portal 18](#_Toc13397)

[1.1 Setup and Configuration changes 18](#_Toc27262)

[1.2 Super Admin 19](#_Toc4516)

[1.3 Tenant Admin 20](#_Toc18239)

[2. Android Application 22](#_Toc19834)

# Introduction

## About this Guide

The purpose of this user guide is to assist developers in understanding and setting up the Mobiliya Fleet Management solution. It is a step-by-step walkthrough the setup process & usage guidelines of the solution.

## About Mobiliya Fleet Management Solution

The complete solution consists of following components,

* Vehicle (Truck/Car)
* OBD/J1939 Dongle
* Mobile Application
* Azure Cloud Application

Commercial Vehicles supporting J1939 protocol will be supported by the solution. Cars supporting OBD Protocol will be supported by the solution.

A user will connect Dongle to vehicle which will retrieve vehicle diagnostic information and forward this information to Mobile application over Bluetooth. Mobile application later on will forward this information to cloud. Cloud application further performs detailed analysis of a given data and provides different reports to stakeholder/user.

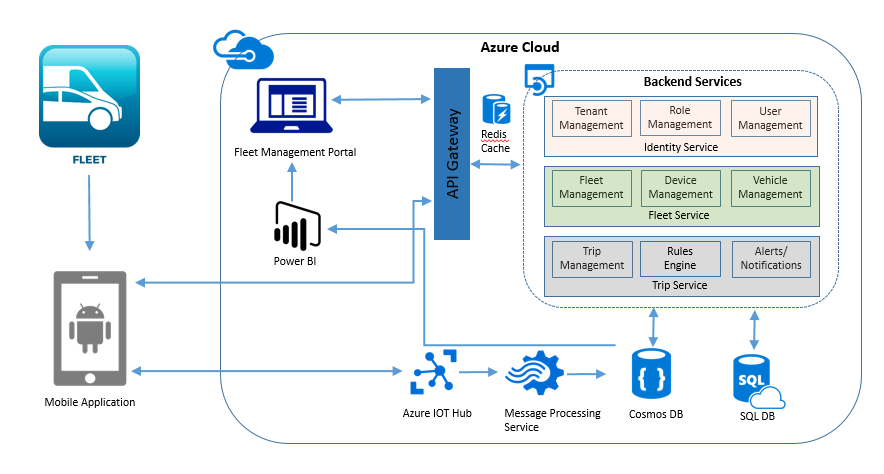
There are 4 types of user actively involved in this system.

* Super Admin (A person who has complete access to system.)
* Tenant Admin
* Fleet Admin
* Driver

# System Architecture

The architecture comprises of two main components,

* Mobile Application
* Cloud Application



## Mobile App Components

Mobile application shall be developed using Android Native Platform. This mobile application shall support Multi-Layer architecture for easy protocol integration.

Mobile application communicates with Dongle using Bluetooth. Mobile application logic is divided in following layers.

* Protocol Dependent Layer
* Protocol independent layer i.e. Business Logic Layer.
* Cloud Communication Layer
* Mobile UI
* Internal Offline Storage

Following Diagram shall provide internal details of the architecture.

Adaptor

BLE

Mobile Application

Protocol Dependent Layer

Protocol independent business logic

Offline Data

Mobile UI

Cloud Communication

Azure IoT Hub

HTTPS

J1939 Dongle

OBD Dongle

Dongle Dependent Lib

(OBD)

Dongle Dependent Lib

(J1939)

* Dongle will communicate with Protocol dependent layer. This will be a configurable interface to support more than 1 dongle. Protocol dependent layer shall encapsulate protocol/Dongle specific implementation details.
* Protocol Independent business logic is intermediate layer which will communicate with Protocol dependent layer, Local Database, Cloud API and Mobile UI.
* Maintaining local database to track trips and vehicle information. This data will be getting synced with server on Azure.

Mobile application shall be developed using,

* Android Native Platform
* SQLite Database
* Azure IoT client

## Cloud Application Components

Cloud application uses shared multi-tenant architecture; where Tenant details and authorization details (roles and privileges) are stored in Master database; however Tenant specific information is stored inside separate Tenant specific databases. Following diagrams provide detailed view of different component used in cloud application.

Mobile 1... N

Web UI

Azure IoT Hub

HTTPS Communication

Azure Backend App service

Master Shared Database

Tenant 1

Data

Storage

Tenant SQL DB

Tenant Blob Storage

Tenant... N

Data

Storage

Tenant SQL DB

Tenant Blob Storage

Tenant Report Ready Data

Tenant Report Ready Data

PowerBI Reports

CLOUD APP

Stream Analytics

There are two active users of cloud application.

1. Web application interface
2. Mobile application.

Web application shall use HTTPS based communication channels and invoke different REST APIs provided by Azure App Service (backend services). Azure App Service shall hold the backend business logic part.

Web application (Fleet Management Portal) is also deployed as Azure App Service and shall be implemented using AngularJS and Bootstrap technologies. It will be a responsive web application and will be fitted to most of desktop/laptop resolution.

The Mobile application shall communicate with Backend Azure Service using two different ways,

1. Azure IoT Hub
2. HTTPS REST API

The Azure IoT hub is used to receive dongle/device data along with Mobile GPS location and forward it to backend Azure App service after every one minute (Send duration will be configurable). Azure backend service internally detects tenant and connect with desired tenant and store the tenant specific data in Tenant database.

The Tenant Specific Report Ready Azure SQL/Cosmos data will be consumed by PowerBI reports.

# Intended Audience

This guide is intended for the users who want to use the system. The guide explains different flows for following user roles:

1. Super Admin

* By default, super admin user is created in system.
* Super admin can manage user accounts.

1. Tenant Admin

* Tenant admin is added by super admin.
* Each company which registers have one tenant admin who can manage delegate users.
* Tenant admin can also manage fleets, vehicles and able to view reports.

1. Fleet Admin

* Fleet admin is added by tenant.
* Fleet admin can add, update or delete drivers.
* Fleet admin also be able to remove vehicles from his/her fleet.

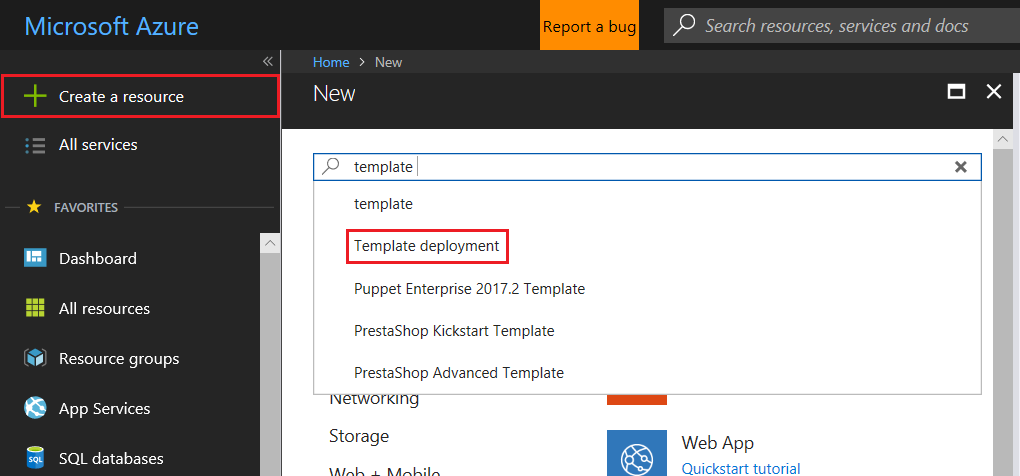
1. Driver

* Drivers can only login to Android app and are not allowed to login on web portal.

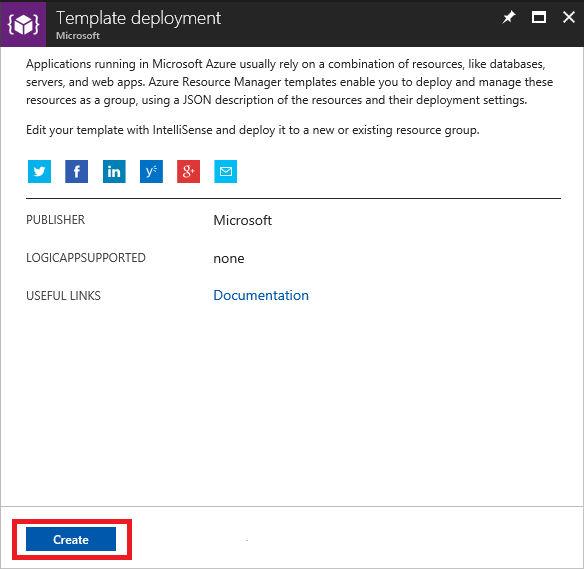
# Setup

Follow below steps for deployment of resources:

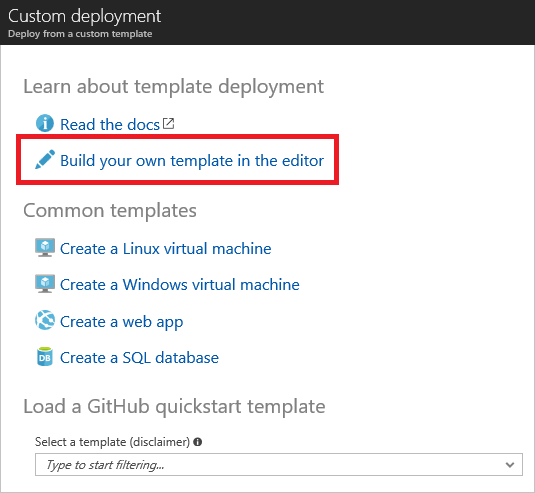
* Clone repository ‘MobiliyaFleetARMTemplate’.
* Create a resource group that serves as the container for the deployed resources.
* Login to the Azure portal (https://portal.azure.com) and select the appropriate subscription if it is not selected by default.
* To deploy a customized template through the portal, select Create a resource, and search for Template Deployment until you can select it from the options



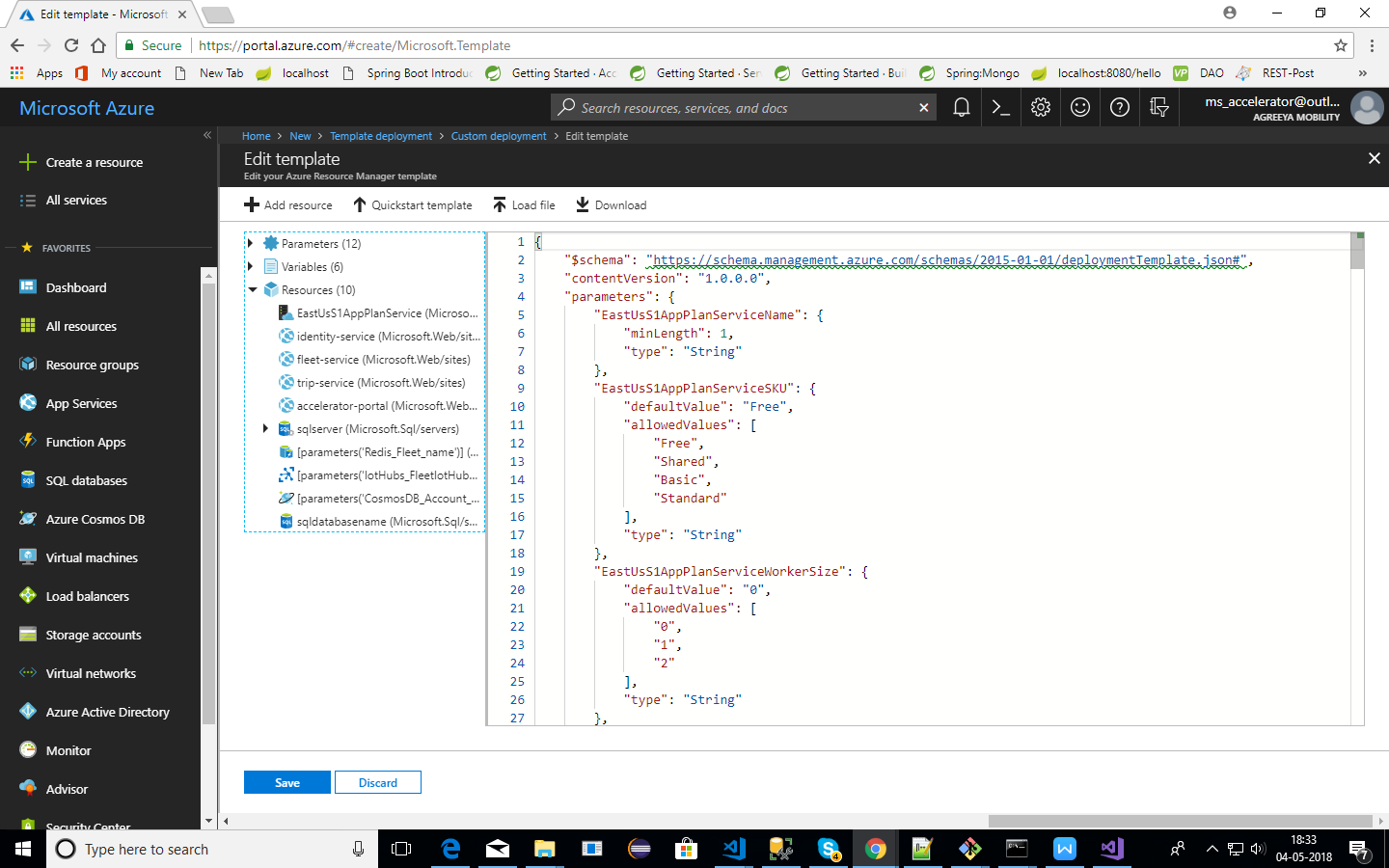
* Select Create



* Select build your own template in the editor.



* Click on load file. Upload deployment template file (azuredeploy.json) from your cloned project.



* Once the file is upload, a form appears which accepts some values before the actual deployment.

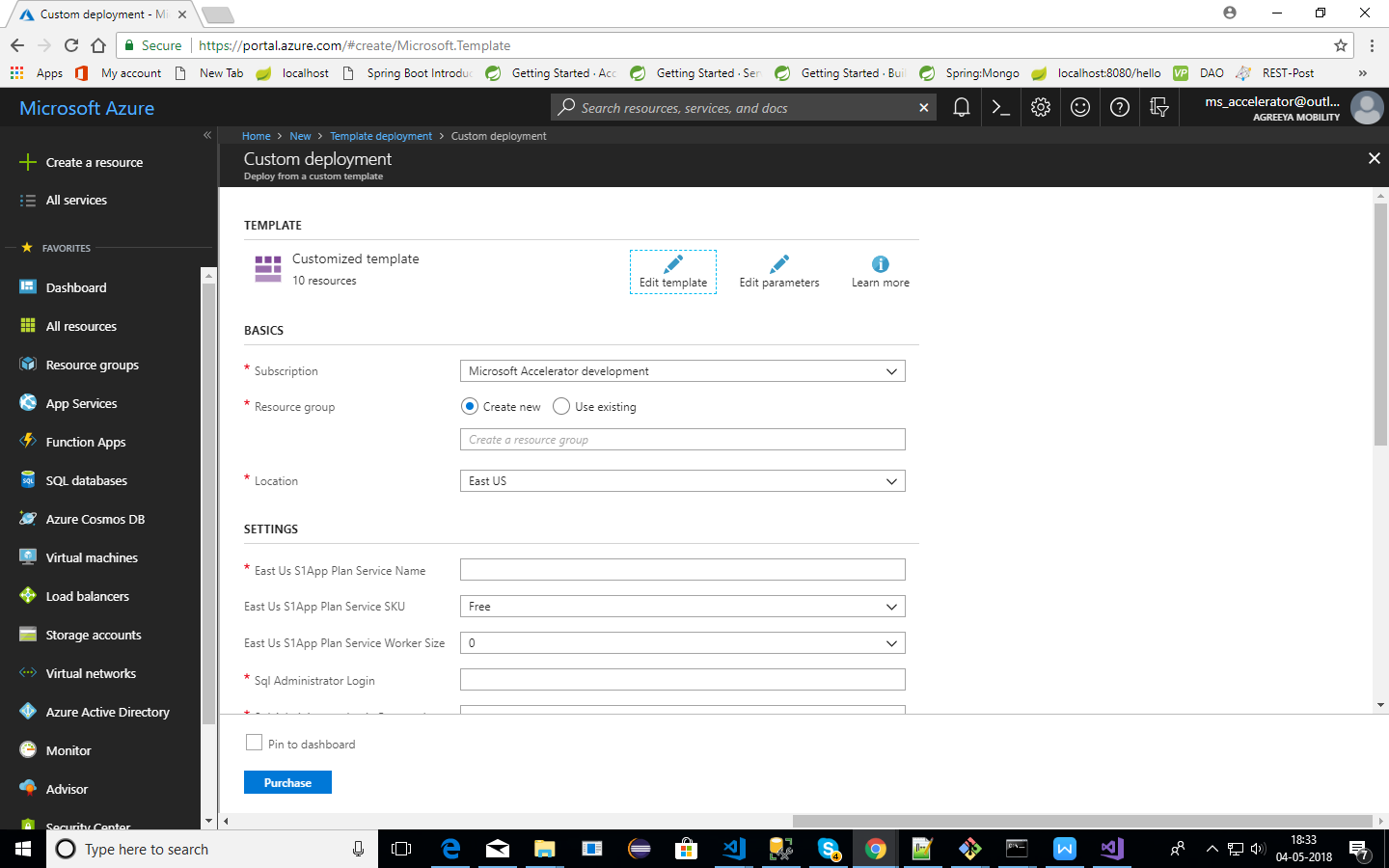
Fill in all the required details using the following guidelines.

Preferable use small case characters.

\* \*\*Do Not\*\* use special symbols.

\* Use the \*\*i\*\* image symbol as a guidance for filling in the data.

\* Give special preference to fields marked with \*\*Has to be unique\*\*.



\* Once, all the fields have been filled, accept the licensing terms and hit \*\*Purchase\*\*.

\* The deployment would take some time; you can have a cup of coffee till then.

\* Once you get a notification of successful deployment of resources from Azure, you can proceed with the further steps.

\* Open the web site and follow the pending steps and configurations

* Make configuration changes in your cloned code (NodeJS backend and AngularJS frontend code) according to deployed resources endpoints and URLs.

Updated following connection information in prod environment of config file to connect to the Azure Resources.

4.1 Identity Service:

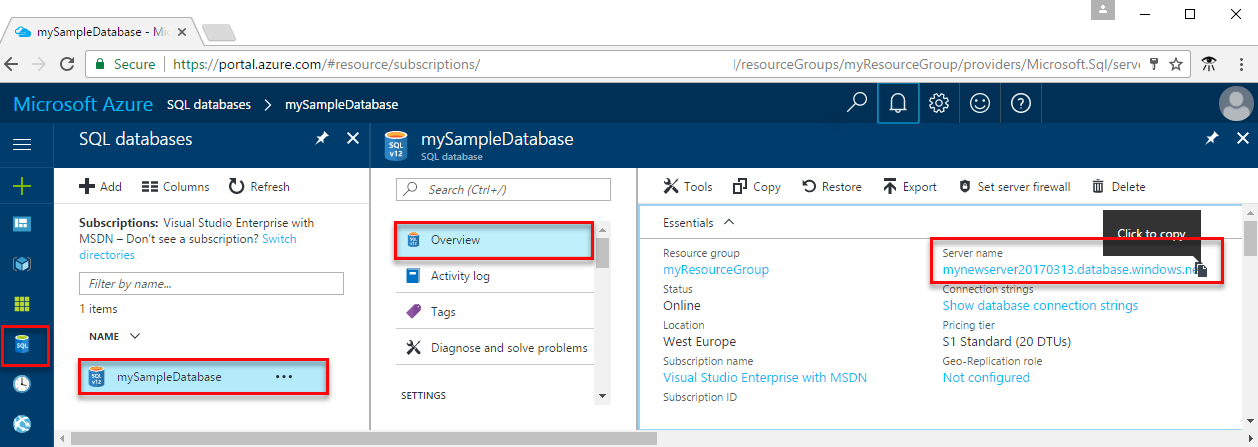
* To update SQL Server Connection you need to follow below steps:

4.1.1 Login into the Azure Portal.

4.1.2 Select SQL Database and go to overview page.

4.1.3 On the overview page,copy server link as shown in below image and

update dbHost parameter with copied link in config file.



4.1.4 Replace the database connection parameter value with the appropriate

values for your server, database, user, and password.

* To update Redis host name and access key you need to follow below steps:

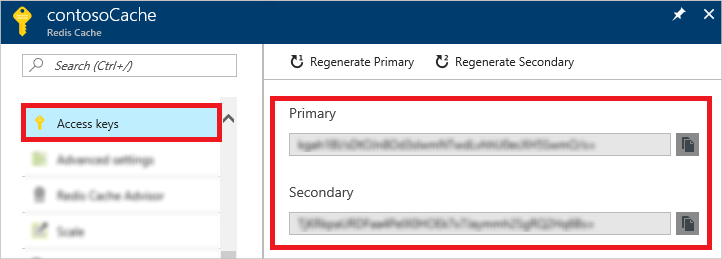
4.1.1 Login into the Azure Portal.

4.1.2 Select Redis and go to overview page.

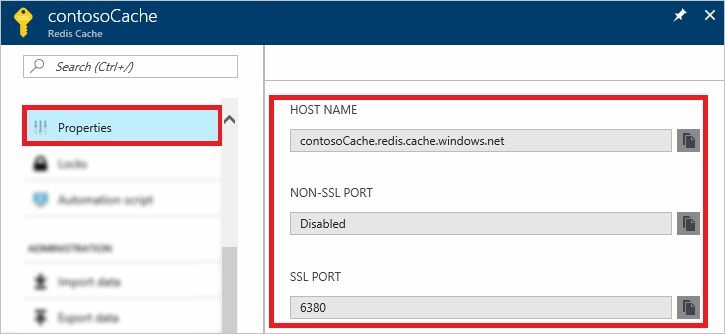
4.1.3 On the overview page, to get auth pass key click on Access Key option as

shown in below image and copy password value from Primary Connection

String and update auth\_pass parameter in config with copied value.



To get Host name click on properties.



4.1.4 Replace the redis connection parameter value with the appropriate values

for your server, port and access key(auth\_pass).

4.2 Fleet Service:

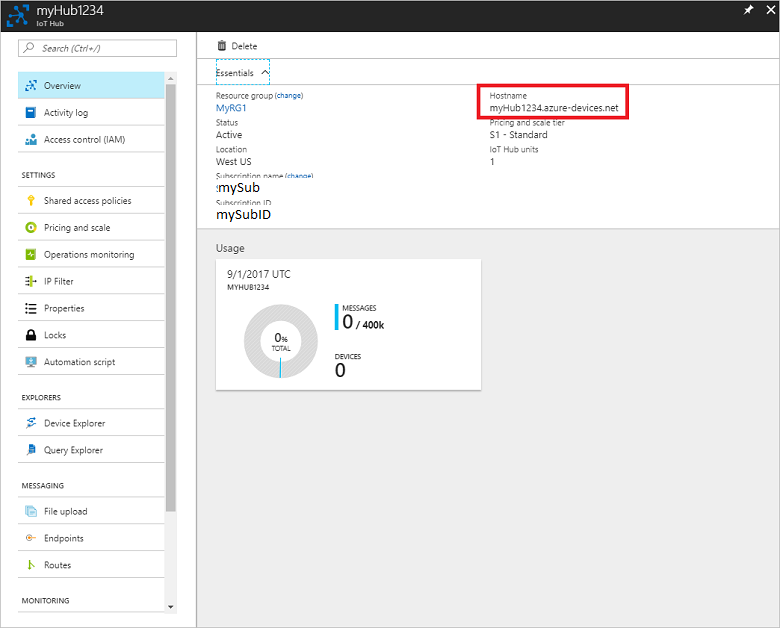
* To update IoTHUB Connection parameter,you need to follow below steps:

4.2.1 Login into the Azure Portal.

4.2.2 Select your IoTHUB resource and go to overview page.

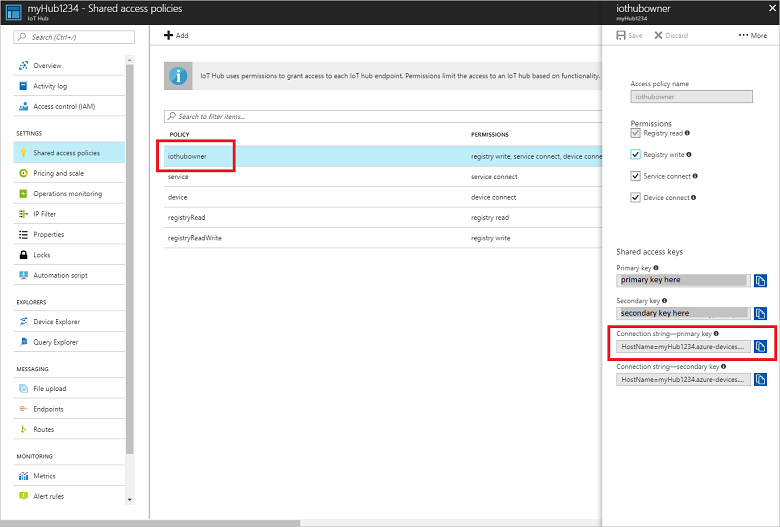
4.2.3 On the overview page,to get Host name click on Hostname option as

Shown in below:



To get connection string click on Shared access policies and copy primary

connection string.



4.2.4 Replace the IoTHUB connection string parameter value with the

copied connection string into config file.

Note: To update SQL Server and Redis Connection in fleet service,please see steps explained in identity service(4.1).

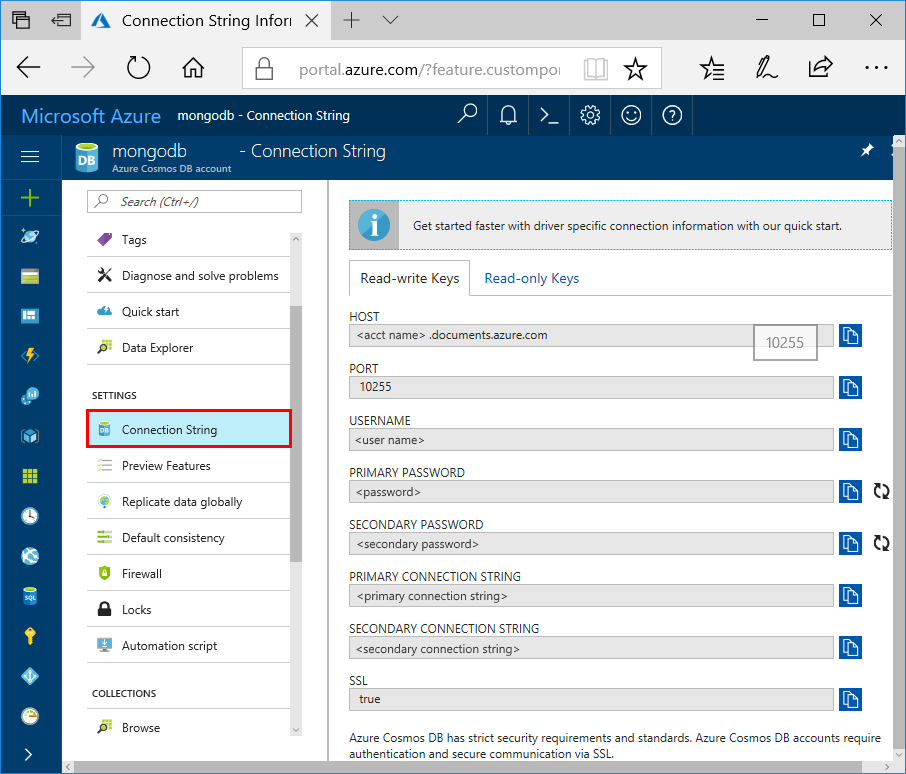
4.3 Trip-service:

* To update Cosmos DB Connection string ,you need to follow below steps:

4.3.1 Login into the Azure Portal.

4.3.2 Select Azure Cosmos Database Account and go to overview page.

4.3.3 On the overview page,to get connection string click on Connection String .



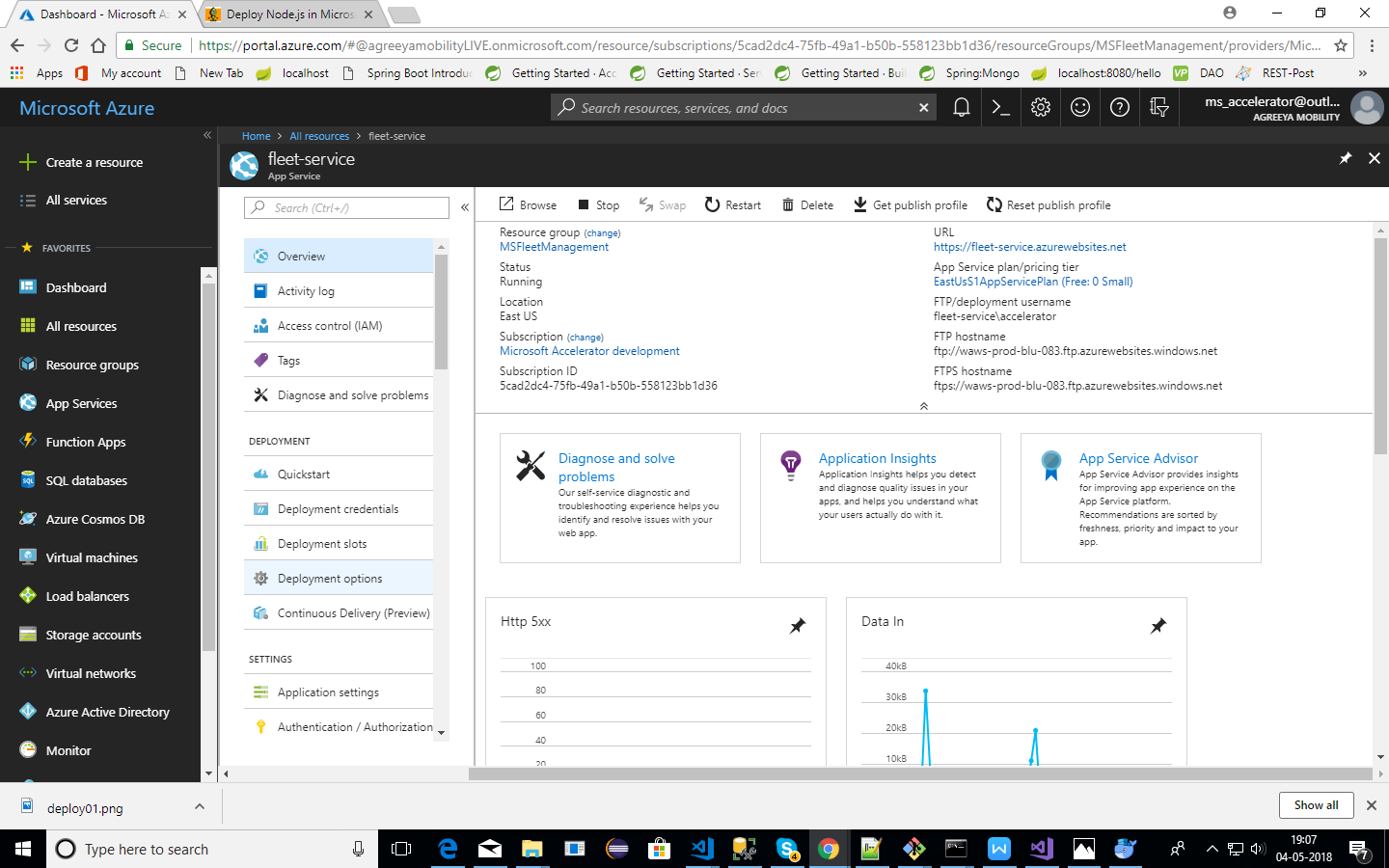
Copy the primary connection string up to only port as shown in below

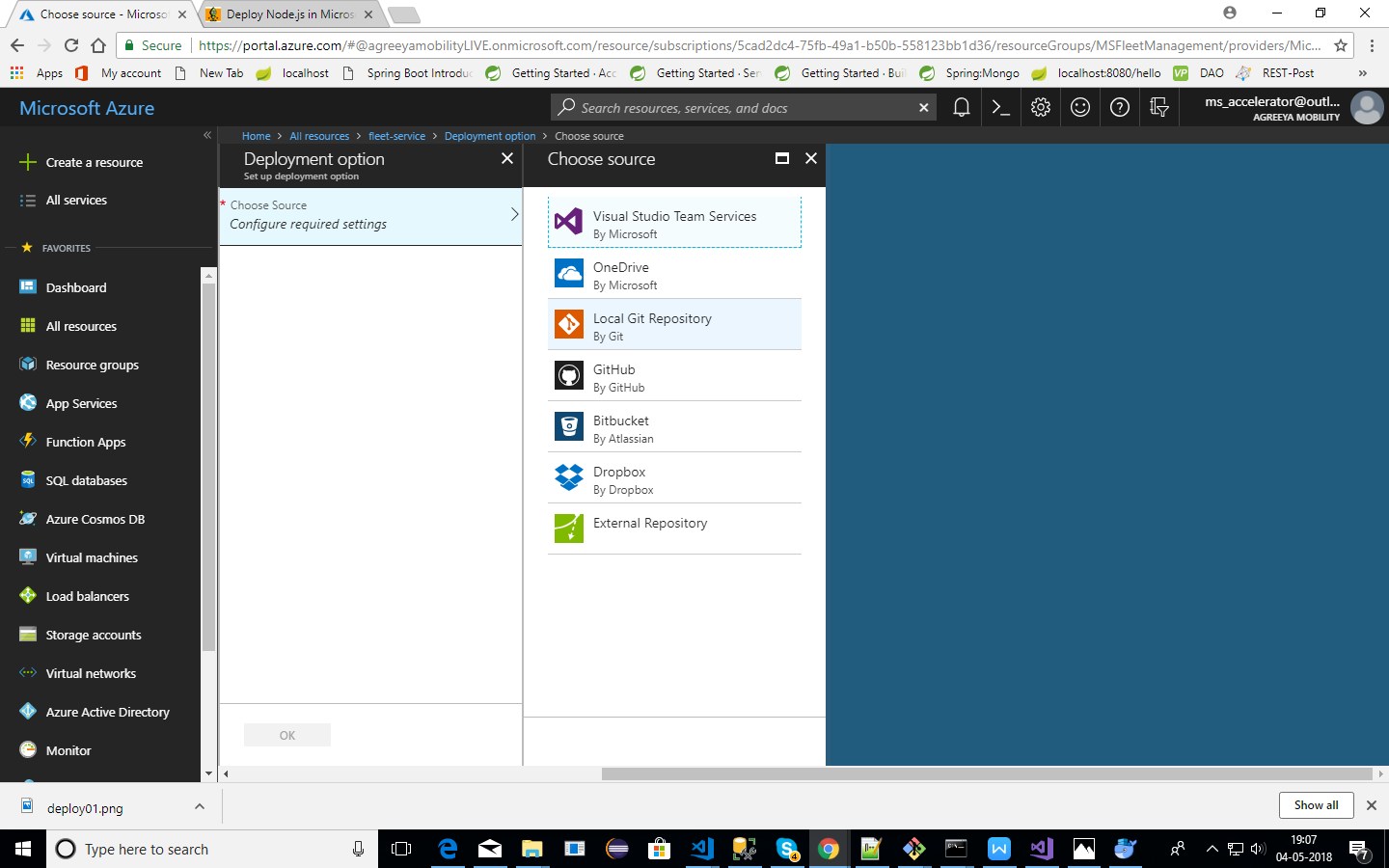
format and paste it into your config file.

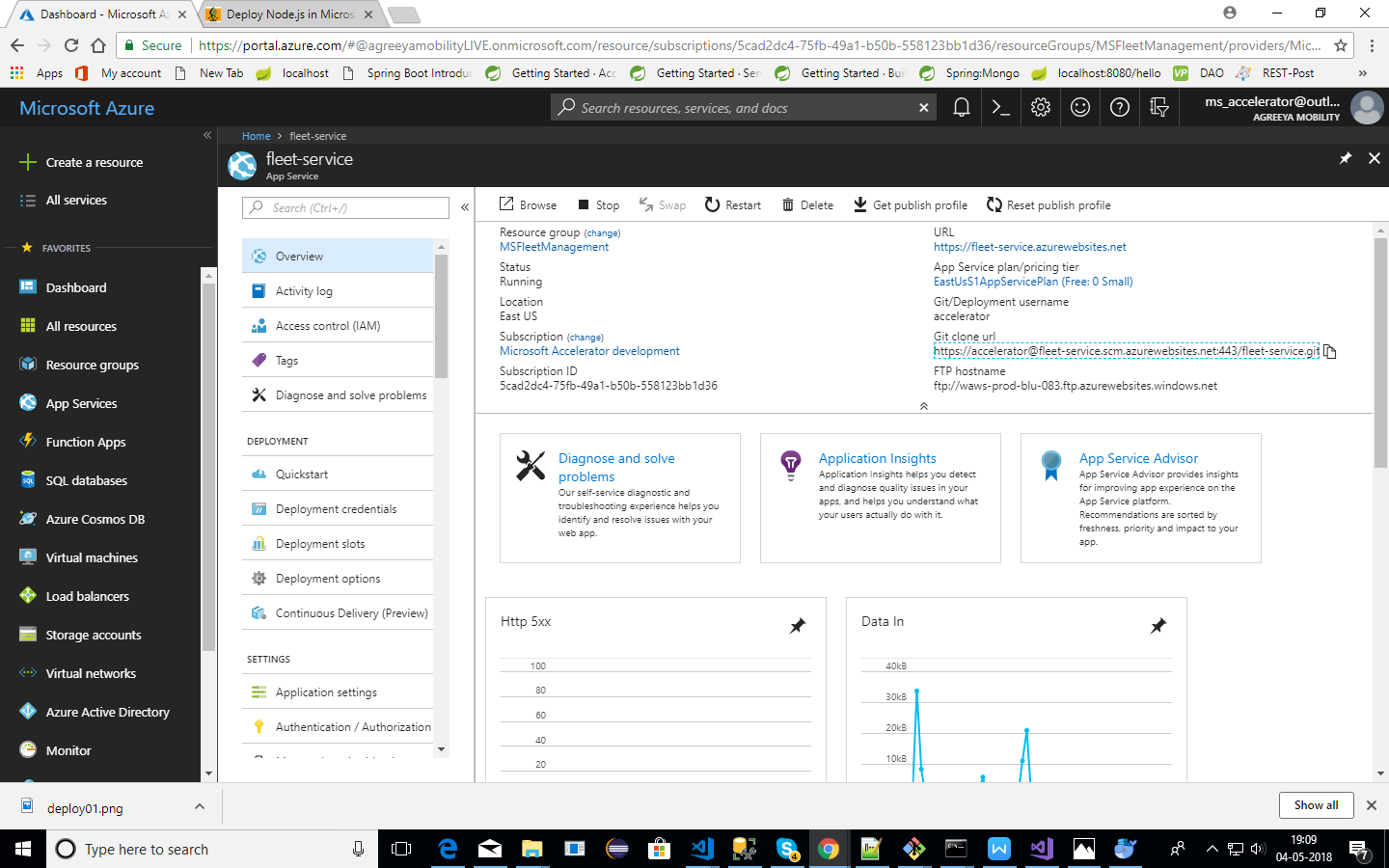
mongodb://username:password@host:port/

Note: To update Redis Connection in trip service,please see steps explained in identity service(4.1) and for iothub connection string see steps explained in fleet service(4.2).

* Once resources are created, click on app service and select Deployment Options



* Click on choose resource and select Local Git Repository option as shown below:
* Copy the git URL and clone the repository.



* Copy your code in that repository and push the changes. This will deploy the changes in the app services.

# User Instructions

## Web Portal

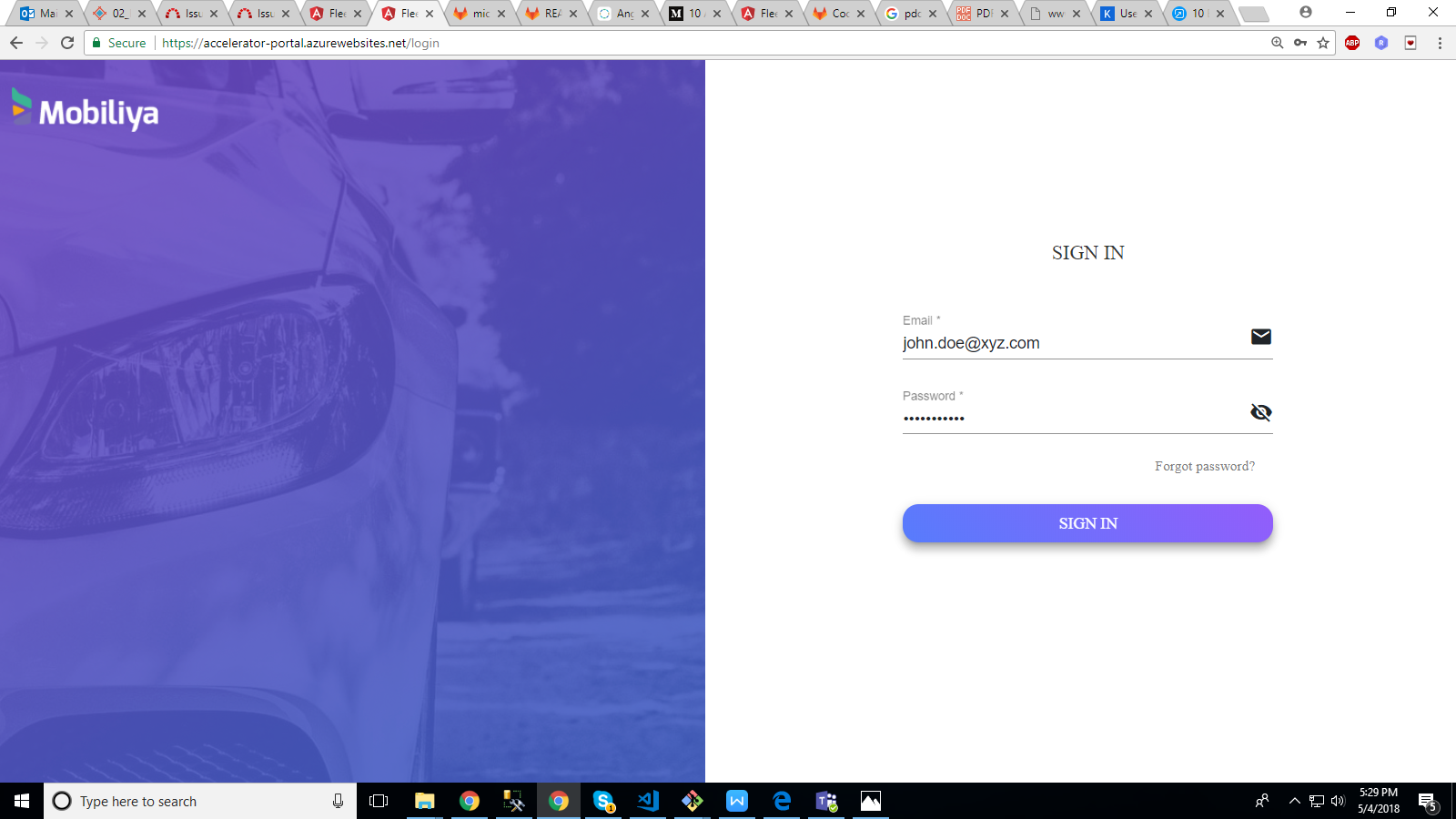
### 1.1 Setup and Configuration changes

1. ***Prerequisites:*** You need to have Git and [Node.js](https://nodejs.org/) installed.
2. ***Clone the repository :*** You have to clone the repository “MobiliyaFleetWebPortal”. (https://github.com/MobiliyaTechnologies/MobiliyaFleetWebPortal.git)
3. ***Environment/config changes*:** This step assumes that you have deployed backend services and those are up and running.   
    You need to change configuration URLs in ‘environment.ts’ file (located at /src/environments/environment.ts), change SERVICE\_URL values.   
   USER(line no 5) : set identity service url  
   FLEET(line no 6): set fleet service url   
   TRIP(line no 7): set trip service url
4. ***Build and Run:*** i. Install the npm packages described in package.json  
    *npm install*ii. You can run the application using below command  
    *ng serve*Application default runs on 4200 port. Open *<https://localhost:4200>* in browser.

### 1.2 Super Admin

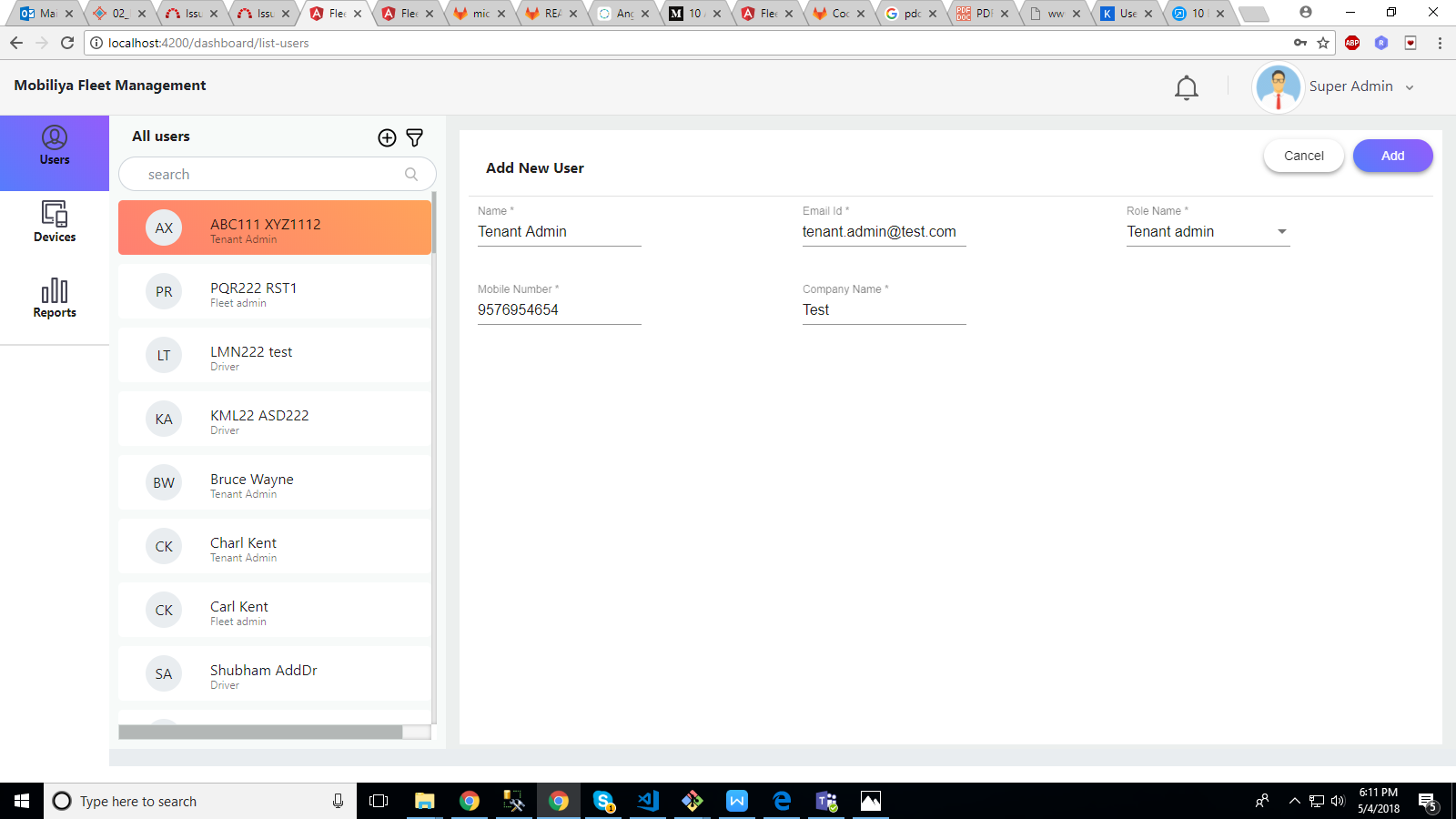
#### Login

Super admin account is created by default. You need to open accelerator portal URL in your browser. And enter your username and password. And click on Sign In button.



#### Add User

On login super admin is directed to user-management page. Super admin can add delegate Users. You can click on add User icon. And add tenant admin.



#### Devices

Super admin can add, edit or delete devices. Devices are nothing but dongle which is associated with vehicle.

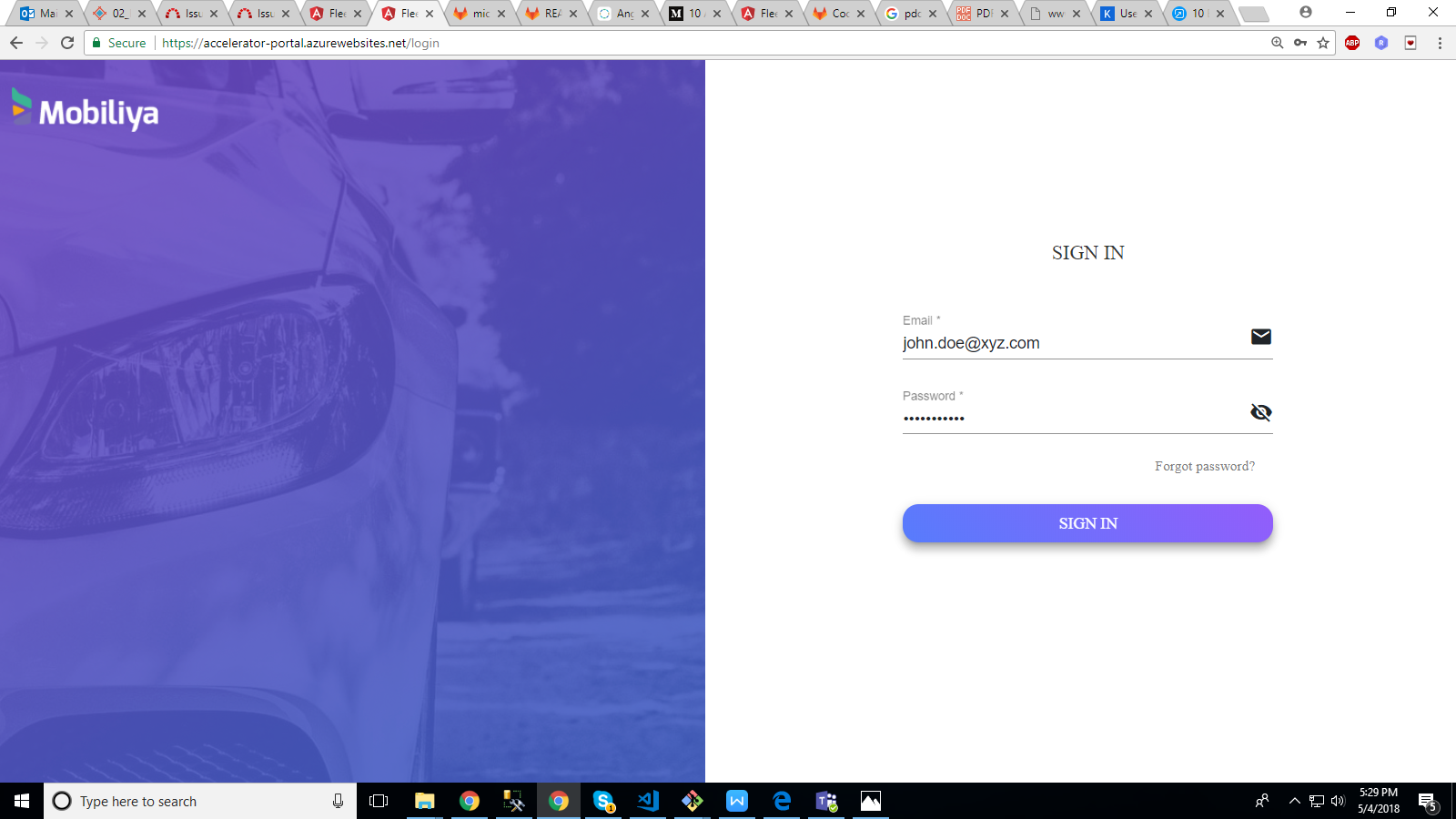
#### Reports

Super admin can view reports. Reports are generated in powerbi. Reports are based on vehicle that is selected. Reports include maximum of one week data where date can be selected.

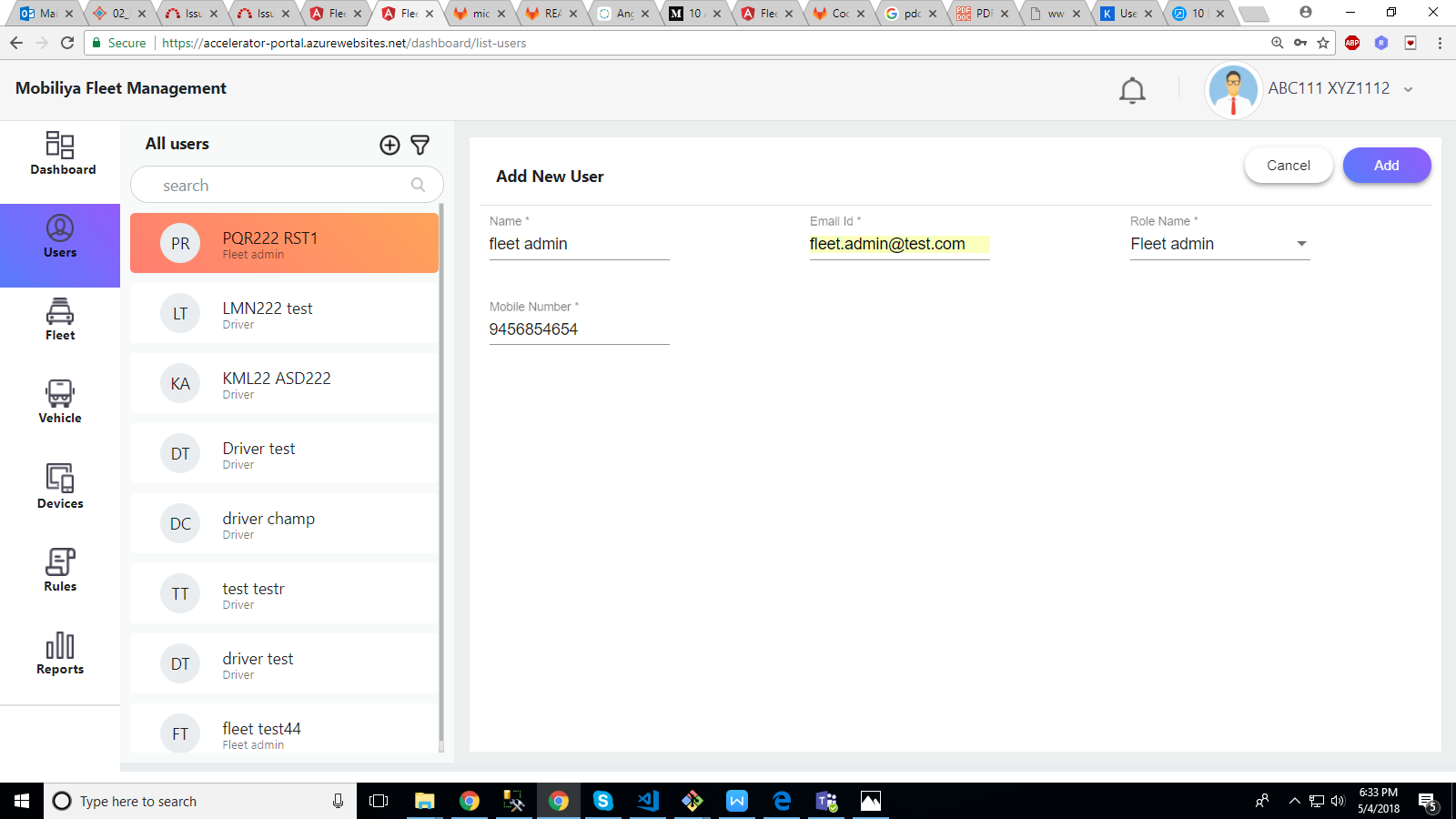
### 1.3 Tenant Admin

#### Login

Tenant Admin can login with the credentials used during sign up. He needs to open accelerator portal URL in your browser. And enter your username and password. And click on Sign In button.



#### Add User

On login tenant admin is directed to dashboard page. Tenant admin can add delegate Users. You can click on add User icon. And add fleet admin.

#### Fleet

Tenant admin can add, edit or delete fleets.

#### Vehicles

Tenant admin can add, edit or delete vehicles. This section also contains trip information of vehicles.

#### Devices

Super admin can add, edit or delete devices. Devices are nothing but dongle which is associated with vehicle.

#### Reports

Tenant admin can view reports. Reports are generated in powerbi. Reports are based on vehicle that is selected. Reports include maximum of one week data where date can be selected.

#### Rules

Tenant admin can create speeding or Geofence rules and assign it to drivers within a fleet.

## Android Application

#### Configuration changes

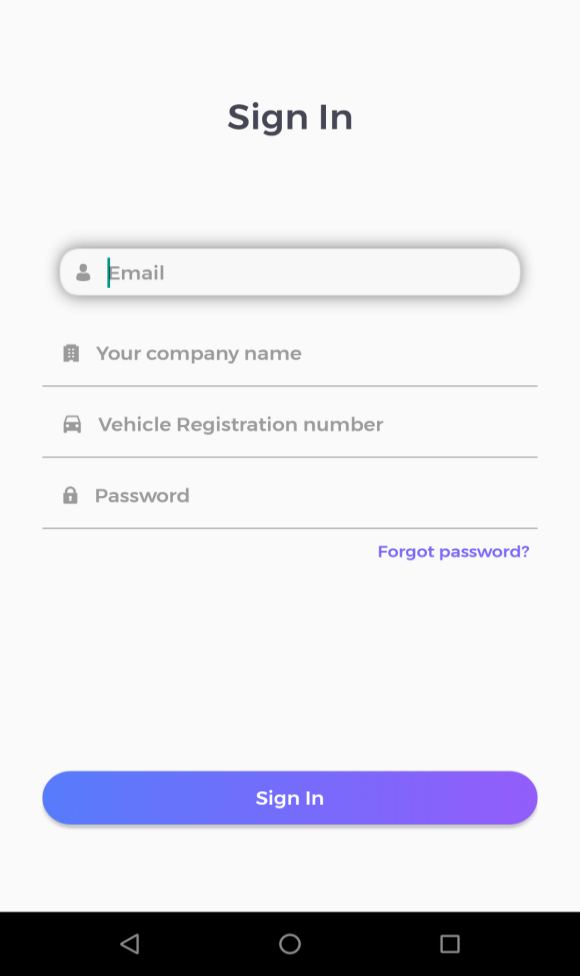
When IOTHub deployment is done then user has to change connection string in the android code file Constants.java, line number 14 “IOT\_CONNSTRING”.

When URL for REST call changes, user has to change below strings from Constants.java file

“IDENTITY\_DEV\_URL” Line no 24,

”FLEET\_DEV\_URL”, Line no 25,

“TRIP\_DEV\_URL” Line no 30



#### Driver Sign In

User can Sign In from here. User can change a password using Forgot Password option.

#### Screenshot_20180504-115959Forgot Password

The Forgot Password option can be accessed from the Sign In Screen. User can change password from here in case user cannot recall the existing password.

#### Screenshot_20180502-174014Dashboard

After successful login and connection with dongle, driver is redirected to dashboard where he can manage trips and view his own ratings, vehicle health.